Predicting shop floor occupancy – Proctor Guide

# Overview

Contoso Corporation is a construction company building shop floors for many manufacturers across the United States. They are interested in incorporating sustainability into their shop floor design as they embark on smart building development. To support this effort, Contoso has invested in IoT technology, installing sensors on 51 of their shop floors. As a launching ground for their sustainability journey, Contoso needs to understand the data collected by their sensors and the factors influenced by the occupancy rate. Using this knowledge, they want to regulate temperature, humidity, light, and monitor CO2 levels on their shop floors.

**This hackathon enables attendees** to access the data Contoso has collected from their sensors to predict if a room does or does not have any occupants. The dataset contains a week’s worth of records or each of the 51 shop floors. This data includes information about CO2 concentration, humidity, room temperature, and luminosity.

The dataset has been obfuscated to prevent any leak of IP or identities from prior analysis, and thus the column variables will be general in nature.

**During the “hacking” attendees will focus on** understanding the data, searching for trends, exploring correlations, the characteristics and impact between CO2, temperature, humidity, light and occupancy rate. Hackers will determine which of these collected features could potentially help in determining if a shop floor is occupied.

**This hackathon enables attendees to** help identify which type of data collected could prove most beneficial in the development of smart buildings.

**This hackathon simulates a real-world scenario** where a construction company is striving to incorporate sustainability in future shop floor designs.

**By the end of the hackathon**, attendees will have built a classification model to predict if a shop floor is occupied.

The main goal for Contoso Corporation is to incorporate sustainability to design smart Carbon and Energy efficient buildings. The challenge is to analyze and interpret data collected and understand the factors influenced by occupancy rate for Contoso.

Once all hackathon challenges are completed, you should be able to:

* Create an Azure Machine Learning workspace
* Collect and process the measurements from the shop floor sensors for data exploration
* Identify the occupancy of the floor with the help of a predictive model
* Understand the features impacted by the shop floor occupancy

# Prerequisites

In these challenges, you will provision an Azure Machine Learning workspace and you will then use the Compute Instance to explore data interactively.

## Prerequisites for Tools

To complete the challenges, you will need to have background knowledge of the following:

* **Language:** Python
  + **Tools:** Azure Machine Learning Studio (Auto ML, Designer can also be used for this challenge but not necessary); Jupyter notebooks
* **Basic Data Science Knowledge:** Data Exploration and Classification modeling building (Relevant prep materials: [Create machine learning models - Learn | Microsoft Docs](https://docs.microsoft.com/en-us/learn/paths/create-machine-learn-models/))

Other basics would be:

* A web browser
* A Microsoft account
* A Microsoft Azure subscription – that would be provided
* A Windows, Linux, or Mac OS X computer
* The challenge files for this course

## Setting up the system

To complete the challenges, you will need the following:

A Microsoft Azure subscription. If you do not have a subscription, please connect with your proctor who will be able to provide one.

An Azure Machine Learning workspace. If you do not have an Azure Machine Learning workspace in your Azure subscription, follow these steps to create one:

1. Sign into the Azure portal using the Microsoft account associated with your Azure subscription.
2. Select ＋Create a resource, search for Machine Learning, and create a new Machine Learning resource with the following settings:
   1. **Workspace Name**: enter a unique name of your choice
   2. **Subscription**: your Azure subscription
   3. **Resource group**: create a new resource group with a unique name
   4. **Location**: choose any available location
3. Wait for your workspace resource to be created, this can take a few minutes. Go to your workspace in the portal, on the Overview page launch Azure Machine Learning studio (or navigate to <https://ml.azure.com>), and sign in using your Microsoft account.
4. In Azure Machine Learning studio, toggle the ☰ icon at the top left to view the various pages in the interface. You can use these pages to manage the resources in your workspace.
5. Create a compute instance, you will need a compute instance in your Azure Machine Learning workspace to run this exercise.

In Azure Machine Learning studio, view the Compute page for your workspace (under Manage). On the Compute Instances tab, if you already have a compute instance, start it; otherwise create a new compute instance with the following settings:

1. **Virtual Machine type**: CPU
2. **Virtual Machine size**: Standard\_DS11\_v2
3. **Compute name**: enter a unique name
4. Wait for the compute instance to start, this may take a few minutes

## Links & Resources for Post Learning Recommendations

* + 1. Sustainability Resources:
       - * Sustainability FAQs: [FAQ (sharepoint.com)](https://microsoft.sharepoint.com/sites/sustainability/sitepages/faq.aspx)
         * MSX Content: <https://aka.ms/MSUSSustainability>
* Sustainability Hub: <https://microsoft.sharepoint.com/sites/sustainability/>
* Environmental Priorities- Carbon: <https://microsoft.sharepoint.com/sites/sustainability/SitePages/Program-Carbon.aspx>
* Environmental Priorities-Ecosystems: <https://microsoft.sharepoint.com/sites/sustainability/SitePages/Ecosystems.aspx>
  + 1. Data Science:
       - * Azure Machine Learning: <https://docs.microsoft.com/en-us/azure/machine-learning/>
         * Data Exploration and Model building: [Create machine learning models - Learn | Microsoft Docs](https://docs.microsoft.com/en-us/learn/paths/create-machine-learn-models/)

# Challenges

There are 4 challenges in total in this for you to successfully complete this hackathon, which are listed below:

## Challenge 1: Load the data set to Azure Machine Learning Workspace

A sophisticated data-collection device, the sensor is a crucial and fascinating component of the Internet of Things (IoT). The purpose of sensors is to collect analog data from the physical world and translate it into digital data assets. Sensors are measuring just about any aspect of the physical world. The calibration of sensors allows them to be tailored to application-specific functions. In this dataset, sensors have been calibrated to measure temperature, humidity, CO2 concentration, luminosity and PIR (motion detection) with accuracy. This sensor data is tasked with capturing information relevant to a shop floor design, so the data can be used to make process improvements for the purpose of increasing carbon and energy efficiency in shop floors.

### Objectives

* How is sensor data collected in Azure Blobs?
* How to ingest and wrangle the data to generate insights from it?

### Tasks

* Data Ingestion Task 1: Download the data from Azure Blob Storage
* Data Ingestion Task 2: Read that data into a single dataframe

### Azure Blob Data Storage Details:

* Connection String:
  + "DefaultEndpointsProtocol=https;AccountName=sustainabilityhackathon;AccountKey=Ejq44H9MM9EZj45ly40vT1cHsZmUAjaIRR+KE5jyqRBqZ+QRZYXwB4+0lJNOGQlHQMVSACChVv9n2GovIPl/WA==;EndpointSuffix=core.windows.net"
* Blob Container:
  + csv

**Data Description**There are 5 types of measurements from each shop floor sensor. Data is collected over a period of one week from Friday, August 23, 2013 to Saturday, August 31, 2013.

* co2.csv - Carbon-dioxide concentration (sampled every 5 seconds)
* humidity.csv - humidity (sampled every 5 seconds)
* light.csv - luminosity (sampled every 5 seconds)
* pir.csv - PIR (passive infrared) motion sensor data (sampled every 10 seconds)
* temperature.csv - shop floor temperature (sampled every 5 seconds)

Each dataset has UNIX Epoch – It is the number of seconds that have elapsed since January 1, 1970. You can choose to convert it to datetime.

**Acknowledgement**: It is hereby acknowledged that the data used here was sourced from publicly available files and channels.

**Proctor Notes** There are often many different reasonable ways to approach the same thing in data science problems. The solution below ingests data and aggregates to represent rows as minute averages, though this is not the only reasonable approach. The ideal solution will reconcile the time differences in measurement collection (5 seconds vs. 10 seconds) without omitting data and losing valuable information. Some other tactics participants might try could include creating a dataset at 5 second intervals with the last known measurement inputed for PIR or taking the average of two PIR readings and inputting that value during the 5 second times. This is just one example of how different participants will have different, but equally valid approaches.

**IN [1]**

**import** azureml**.**core

**from** azureml**.**core **import** Workspace

# Load the workspace from the saved config file

ws **=** Workspace**.**from\_config**()**

**print(**'Ready to use Azure ML {} to work with {}'**.format(**azureml**.**core**.**VERSION**,** ws**.**name**))**

**IN [2]**

# Python program to bulk download blob files from azure storage

**import** os

**from** azure**.**storage**.**blob **import** BlobServiceClient**,** BlobClient

**from** azure**.**storage**.**blob **import** ContentSettings**,** ContainerClient

MY\_CONNECTION\_STRING **=** "DefaultEndpointsProtocol=https;AccountName=sustainabilityhackathon;AccountKey=Ejq44H9MM9EZj45ly40vT1cHsZmUAjaIRR+KE5jyqRBqZ+QRZYXwB4+0lJNOGQlHQMVSACChVv9n2GovIPl/WA==;EndpointSuffix=core.windows.net"

# Replace with blob container

MY\_BLOB\_CONTAINER **=** "csv"

# Replace with the local folder where you want files to be downloaded

LOCAL\_BLOB\_PATH **=** "shopfloordata"

**class** **AzureBlobFileDownloader:**

**def** \_\_init\_\_**(**self**):**

**print(**"Intializing AzureBlobFileDownloader"**)**

# Initialize the connection to Azure storage account

self**.**blob\_service\_client **=** BlobServiceClient**.**from\_connection\_string**(**MY\_CONNECTION\_STRING**)**

self**.**my\_container **=** self**.**blob\_service\_client**.**get\_container\_client**(**MY\_BLOB\_CONTAINER**)**

**def** save\_blob**(**self**,**file\_name**,**file\_content**):**

# Get full path to the file

download\_file\_path **=** os**.**path**.**join**(**LOCAL\_BLOB\_PATH**,** file\_name**)**

# For nested blobs, create local path as well!

os**.**makedirs**(**os**.**path**.**dirname**(**download\_file\_path**),** exist\_ok**=True)**

**with** **open(**download\_file\_path**,** "wb"**)** **as** file**:**

file**.**write**(**file\_content**)**

**def** download\_all\_blobs\_in\_container**(**self**):**

my\_blobs **=** self**.**my\_container**.**list\_blobs**()**

**for** blob **in** my\_blobs**:**

**print(**blob**.**name**)**

**bytes** **=** self**.**my\_container**.**get\_blob\_client**(**blob**).**download\_blob**().**readall**()**

self**.**save\_blob**(**blob**.**name**,** **bytes)**

# Initialize class and upload files

azure\_blob\_file\_downloader **=** AzureBlobFileDownloader**()**

azure\_blob\_file\_downloader**.**download\_all\_blobs\_in\_container**()**

**OUT[2]**

# Sample output data set

Intializing AzureBlobFileDownloader

KETI/413/co2.csv

KETI/413/humidity.csv

KETI/413/light.csv

KETI/413/pir.csv

KETI/413/temperature.csv

KETI/415/co2.csv

KETI/415/humidity.csv

KETI/415/light.csv

KETI/415/pir.csv

KETI/415/temperature.csv

KETI/417/co2.csv

KETI/417/humidity.csv

KETI/417/light.csv

KETI/417/pir.csv

KETI/417/temperature.csv

KETI/419/co2.csv

KETI/419/humidity.csv

KETI/419/light.csv

KETI/419/pir.csv

KETI/419/temperature.csv

KETI/421/co2.csv

**IN [3]**

# Read the data directory structure

**import** os

**import** pandas **as** pd

directory\_list **=** **list()**

**for** root**,** dirs**,** files **in** os**.**walk**(**LOCAL\_BLOB\_PATH**,** topdown**=False):**

**for** name **in** dirs**:**

directory\_list**.**append**(**os**.**path**.**join**(**root**,** name**))**

# Get the sub directory list of ShopFloorSensorData folder

**del** directory\_list**[**51**]**

**IN [4]**

# Read data from each folder and combine the data from each folder to get the full data

# set

# Since the data is collected at 5 and 10 second intervals, aggregate the data at a

# minute level for this study

**from** functools **import** reduce

**import** datetime

**import** pandas **as** pd

final\_data **=** pd**.**DataFrame**()**

**for** **dir** **in** directory\_list**:**

light\_path **=** **dir** **+** '/light.csv'

temperature\_path **=** **dir** **+** '/temperature.csv'

co2\_path **=** **dir** **+** '/co2.csv'

pir\_path **=** **dir** **+** '/pir.csv'

humidity\_path **=** **dir** **+** '/humidity.csv'

light\_df **=** pd**.**read\_csv**(**light\_path**,** names**=[**'unix\_time\_light'**,** 'light'**])**

temperature\_df **=** pd**.**read\_csv**(**temperature\_path**,** names**=[**'unix\_time\_temp'**,** 'temperature'**])**

co2\_df **=** pd**.**read\_csv**(**co2\_path**,** names**=[**'unix\_time\_co2'**,** 'co2'**])**

pir\_df **=** pd**.**read\_csv**(**pir\_path**,** names**=[**'unix\_time\_pir'**,** 'pir'**])**

humidity\_df **=** pd**.**read\_csv**(**humidity\_path**,** names**=[**'unix\_time\_humidity'**,** 'humidity'**])**

light\_df**[**'unix\_time\_light'**]** **=** light\_df**[**'unix\_time\_light'**].**apply**(lambda** x**:** datetime**.**datetime**.**fromtimestamp**(**x**))**

light\_df**[**'unix\_time\_light'**]** **=** light\_df**[**'unix\_time\_light'**].**dt**.round(**'1min'**)**

light\_df **=** light\_df**.**groupby**(**pd**.**Grouper**(**key**=**"unix\_time\_light"**)).**mean**()**

temperature\_df**[**'unix\_time\_temp'**]** **=** temperature\_df**[**'unix\_time\_temp'**].**apply**(lambda** x**:** datetime**.**datetime**.**fromtimestamp**(**x**))**

temperature\_df**[**'unix\_time\_temp'**]** **=** temperature\_df**[**'unix\_time\_temp'**].**dt**.round(**'1min'**)**

temperature\_df **=** temperature\_df**.**groupby**(**pd**.**Grouper**(**key**=**"unix\_time\_temp"**)).**mean**()**

co2\_df**[**'unix\_time\_co2'**]** **=** co2\_df**[**'unix\_time\_co2'**].**apply**(lambda** x**:** datetime**.**datetime**.**fromtimestamp**(**x**))**

co2\_df**[**'unix\_time\_co2'**]** **=** co2\_df**[**'unix\_time\_co2'**].**dt**.round(**'1min'**)**

co2\_df**=** co2\_df**.**groupby**(**pd**.**Grouper**(**key**=**"unix\_time\_co2"**)).**mean**()**

pir\_df**[**'unix\_time\_pir'**]** **=** pir\_df**[**'unix\_time\_pir'**].**apply**(lambda** x**:** datetime**.**datetime**.**fromtimestamp**(**x**))**

pir\_df**[**'unix\_time\_pir'**]** **=** pir\_df**[**'unix\_time\_pir'**].**dt**.round(**'1min'**)**

pir\_df**=** pir\_df**.**groupby**(**pd**.**Grouper**(**key**=**"unix\_time\_pir"**)).**median**()**

humidity\_df**[**'unix\_time\_humidity'**]** **=** humidity\_df**[**'unix\_time\_humidity'**].**apply**(lambda** x**:** datetime**.**datetime**.**fromtimestamp**(**x**))**

humidity\_df**[**'unix\_time\_humidity'**]** **=** humidity\_df**[**'unix\_time\_humidity'**].**dt**.round(**'1min'**)**

humidity\_df **=** humidity\_df**.**groupby**(**pd**.**Grouper**(**key**=**"unix\_time\_humidity"**)).**mean**()**

data **=** pd**.**concat**([**light\_df**,** temperature\_df**,**co2\_df**,**pir\_df**,**humidity\_df**],** axis**=**1**,** join**=**"outer"**)**

data**[**'ShopFloor'**]** **=** **dir.**lstrip**(**'ShopFloorSensorData/KETI/'**)**

**print(**"Dir {} light\_df: {} temperature\_df: {} co2\_df: {} pir\_df: {} humidity\_df: {}"**.format(dir,len(**light\_df**),len(**temperature\_df**),** **len(**co2\_df**),** **len(**pir\_df**),** **len(**humidity\_df**)))**

final\_data **=** final\_data**.**append**(**data**)**

**OUT[4]**

Dir ShopFloorSensorData/KETI/413 light\_df: 10924 temperature\_df: 10924 co2\_df: 10923 pir\_df: 11995 humidity\_df: 10924

Dir ShopFloorSensorData/KETI/415 light\_df: 10924 temperature\_df: 10924 co2\_df: 11021 pir\_df: 11995 humidity\_df: 10924

Dir ShopFloorSensorData/KETI/417 light\_df: 10923 temperature\_df: 10923 co2\_df: 10924 pir\_df: 11995 humidity\_df: 10923

Dir ShopFloorSensorData/KETI/419 light\_df: 10910 temperature\_df: 10910 co2\_df: 10927 pir\_df: 11995 humidity\_df: 10910

Dir ShopFloorSensorData/KETI/421 light\_df: 10937 temperature\_df: 10937 co2\_df: 10935 pir\_df: 11995 humidity\_df: 10937

**IN [5]**

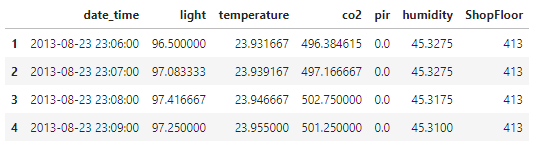
# Sample data

final\_data**.**reset\_index**(**inplace**=True)**

final\_data **=** final\_data**.**rename**(**columns **=** **{**'index'**:**'date\_time'**})**

final\_data**[**1**:**5**]**

**OUT [5]**



**Proctor Notes:** The above code walks through one way this data aggregation can be done. The code you review may look different, or use different methods and variable names, but yield an equally valid output. The ideal solution will reconcile missing data related to the 5 to 10 second interval mismatch and likely appear similar to the data table in the output above (**OUT[5]**).

## Challenge 2: Data Exploration

Data exploration is an approach to understand what is in a dataset and the characteristics of the data. These characteristics can include size or amount of data, completeness of the data, correctness of the data, possible relationships amongst data elements or files/tables in the data. Data Exploration is aimed at understanding the nuances of the data, and defining basic metadata (statistics, structure, relationships) for the data set that can be used in further analysis. Once this initial understanding of the data is had, the data can be pruned or refined by removing unusable parts of the data, correcting poorly formatted elements and defining relevant relationships across datasets.

### Objectives

* What are the ways data can be pruned or refined by removing unusable and poorly formatted data portions?
* How to visualize relevant relationships amongst features across dataset?

### Tasks

1. Is there any missing data? Does data need any imputation? If so, take the suitable data imputation? If so, take the suitable data imputation measures.

2. Do certain days and hours of the day have more occupancy? Any interesting trends you can comment on?

3. Identify correlations between the features. Are there features that are highly correlated to shop floor occupancy rate?

4. What are the characteristics of the features co2, temperature, humidity and light?

5. Which of the features co2, temperature, humidity and light could potentially help in determining if a shop floor is occupied or not?

**NOTE: Once the last part of this challenge has been unlocked, the badge would automatically be issued to all the participants.**

### **Data Exploration challenge 1: Is there any missing data? Does data need any imputation and take the suitable data imputation measures if needed.**

**IN [6]**

# Check for isnull in the data

final\_data**.**isnull**().sum()/len(**final\_data**)** # There is quite a bit of missing data

# Insight:

# After data aggregation at minute level, PIR has approximately 2% missing data and light, temperature, co2 and humidity has around 8% missing data

**OUT [6]**

date\_time 0.000000

light 0.089731

temperature 0.089731

co2 0.089440

pir 0.023299

humidity 0.089731

ShopFloor 0.000000

dtype: float64

**IN [7]**

# Since the percent missing data is small, just delete the missing data rows

# Remove rows where pir and rest of the variables is null

final\_data **=** final\_data**[**final\_data**[**'pir'**].**notna**()]**

final\_data **=** final\_data**[**final\_data**[**'light'**].**notna**()]**

final\_data **=** final\_data**[**final\_data**[**'co2'**].**notna**()]**

# Check the rows that still have no data

final\_data**.**isnull**().sum()**

**OUT [7]**

date\_time 0

light 0

temperature 0

co2 0

pir 0

humidity 0

ShopFloor 0

dtype: int64

### **Data Exploration Challenge 2: Do certain days and hours of the day have more occupancy? Any interesting trends you can comment on?**

**Proctor Notes:** While the examples below focus on plotting the relationship between dates and room occupancy and between hour of the day and room occupancy, participants may analyze additional features related to these features such as the previous day or hour’s level of occupancy and changes in occupancy between periods, among many other relationships.

**IN [8]**

**import** matplotlib**.**pyplot **as** plt

**import** seaborn **as** sns

**import** numpy **as** np

final\_data**[**'Date'**]** **=** pd**.**to\_datetime**(**final\_data**[**'date\_time'**],**unit**=**'s'**)**

final\_data**[**'Date'**]** **=** final\_data**[**'Date'**].**dt**.**to\_period**(**'D'**)**

occupancy\_freq\_day **=** final\_data**.**groupby**(**by**=[**'ShopFloor'**,** 'Date'**]).**agg**({**'pir'**:** **lambda** x**:** np**.sum(**x**)/len(**x**)\***100**}).**unstack**()**

plt**.**figure**(**figsize**=(**20**,**5**))**

sns**.**heatmap**(**occupancy\_freq\_day**.**T**,** cmap**=**sns**.**color\_palette**(**'Greens'**),** linewidths**=**0.1**,** linecolor**=**'white'**)**

plt**.**xticks**(**rotation**=**'90'**)**

plt**.**title**(**'Shop Floor Occupancy per Day'**)**

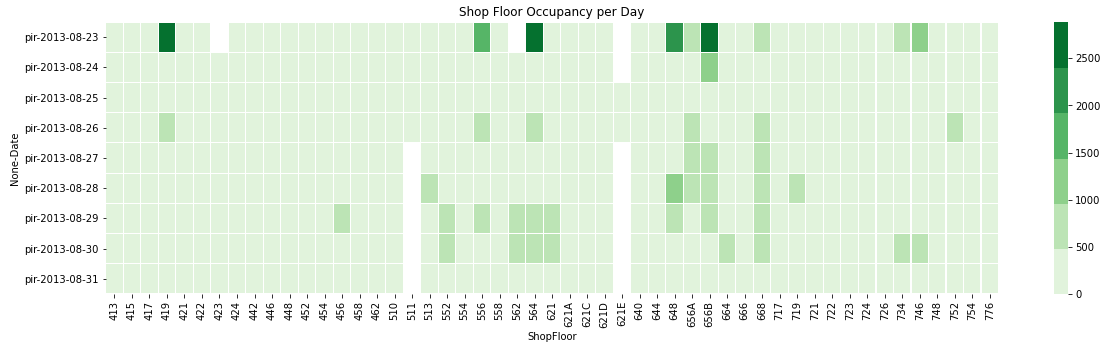
plt**.**show**()**

# Insight:

# Shop floor 419, 564,648, 656b has high occupancy rate on 2013/08/23

# Not much occupancy/no data on 2013/08/29

**OUT [8]**



**IN [9]**

final\_data**[**'Hour'**]** **=** pd**.**to\_datetime**(**final\_data**[**'date\_time'**],**unit**=**'s'**)**

final\_data**[**'Hour'**]** **=** final\_data**[**'Hour'**].**dt**.**hour

occupancy\_freq\_hour**=** final\_data**.**groupby**(**by**=[**'ShopFloor'**,** 'Hour'**]).**agg**({**'pir'**:** **lambda** x**:** np**.sum(**x**)/len(**x**)\***100**}).**unstack**()**

plt**.**figure**(**figsize**=(**20**,**5**))**

sns**.**heatmap**(**occupancy\_freq\_hour**.**T**,** cmap**=**sns**.**color\_palette**(**'Greens'**),** linewidths**=**0.1**,** linecolor**=**'white'**)**

plt**.**xticks**(**rotation**=**'90'**)**

plt**.**title**(**'Shop Floor Occupancy by hour of the day'**)**

plt**.**show**()**

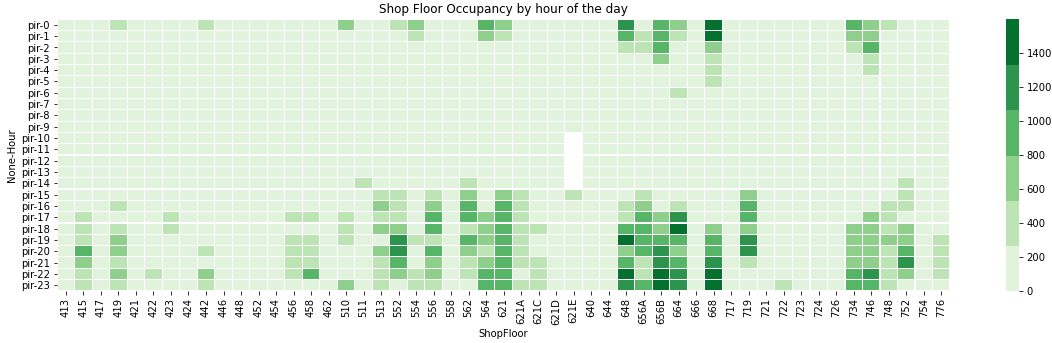
# Insight:

# Shopfloors have higher occupancy towards late afternoon and night

# Some shop floors also have some occupancy during the late hours. May indicate workers doing second sift until the wee hours

# 4XX shop floors have relatively lower occupancy and 6XX shop floors have relatively higher occupancy

**OUT [9]**



**IN [10]**

# Based on the previous analysis:

# Reduce the number of sensors based on shopfloor type

# There are 4 kinds, 4XX, 5XX, 6XX and 7XX, of shop floor sensors

final\_data**[**'ShopFloor\_Type'**]** **=** final\_data**[**'ShopFloor'**].str[**0**]**

**Proctor Notes:** These are just a sample of observations that participants could make based on the datetimes. There are many other possible observations that participants could make including the previous occupancy rate, the change in occupancy rate over time, or change of other metrics over time.

### **Data Exploration Challenge 3: Identify correlations between the features. Are there feature that are highly correlated to shop floor occupancy rate?**

**IN [11]**

numerical\_features **=** **[**'light'**,** 'temperature'**,** 'co2'**,** 'pir'**,** 'humidity'**,**'Hour'**]**

corr **=** final\_data**[**numerical\_features**].**corr**(**'spearman'**)**

corr**.**style**.**background\_gradient**(**cmap**=**'coolwarm'**)**

# Insight:

# temperature and humidity have high negative correlation as expected

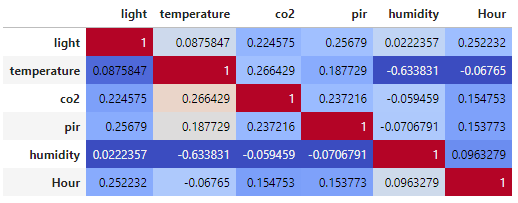
# Hour and light has mild correlation as expected

# C02 and temperature have mild positive correlation

# Co2 and light have mild positive correlation

# pir has mild positive correlation with temperature, light, co2 and hour

**OUT [11]**



### **Data Exploration Challenge 4: What are the characteristics of the features CO2, temperature, humidity and light?**

**IN [12]**

features **=** **[**'light'**,** 'temperature'**,** 'co2'**,** 'humidity'**]**

final\_data**[**features**].**agg**([**'skew'**,** 'kurtosis'**]).**transpose**()**

#Insight:

# When you look at the features without considering the shop floor sensors -

# temperature, light are highly skewed

# co2, humidity are fairly symmetrical. The high kurtosis values for light,

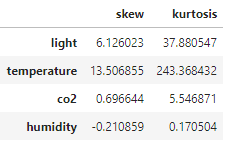
# ‘temperature, & CO2 show that the outlier values in the distribution of

# those features are more extreme than a normal distribution. Humidity has a

# kurtosis very close to 0, so that alongside its skew shows that it has

# essentially a normal distribution.

**OUT [12]**



**IN [13]**

**import** seaborn **as** sns

final\_data**[**'light\_log'**]** **=** final\_data**[**'light'**].**apply**(lambda** x**:** np**.**log**(**x**+**1**))**

# The purpose of the log transformation for these variables is to mitigate

the impact of high outliers on the model’s predictions

fig **=** plt**.**gcf**()**

sns**.**violinplot**(**x**=** "light\_log"**,** y**=**"ShopFloor\_Type"**,** data**=**final\_data**)**

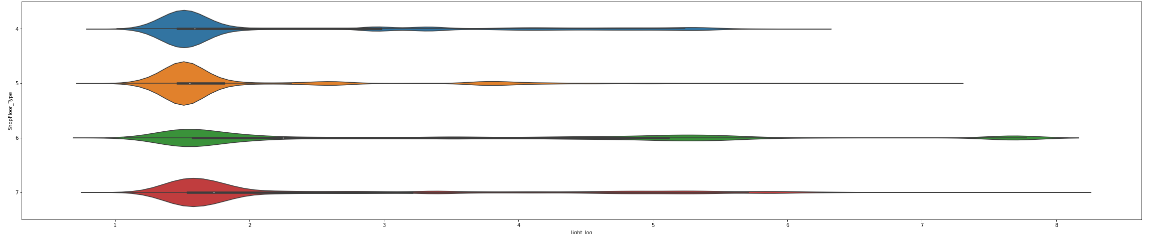
fig**.**set\_size\_inches**(**40**,** 8**)**

#Insight

# Multi model right skewed distributions with large outliers

# Shop Floors 6XX has the height median luminosity

**OUT [13]**



**IN [14]**

# Temperature distribution

final\_data**[**'temperature\_log'**]** **=** final\_data**[**'temperature'**].**apply**(lambda** x**:** np**.**log**(**x**+**1**))**

fig **=** plt**.**gcf**()**

sns**.**violinplot**(**x**=** "temperature\_log"**,** y**=**"ShopFloor\_Type"**,** data**=**final\_data**)**

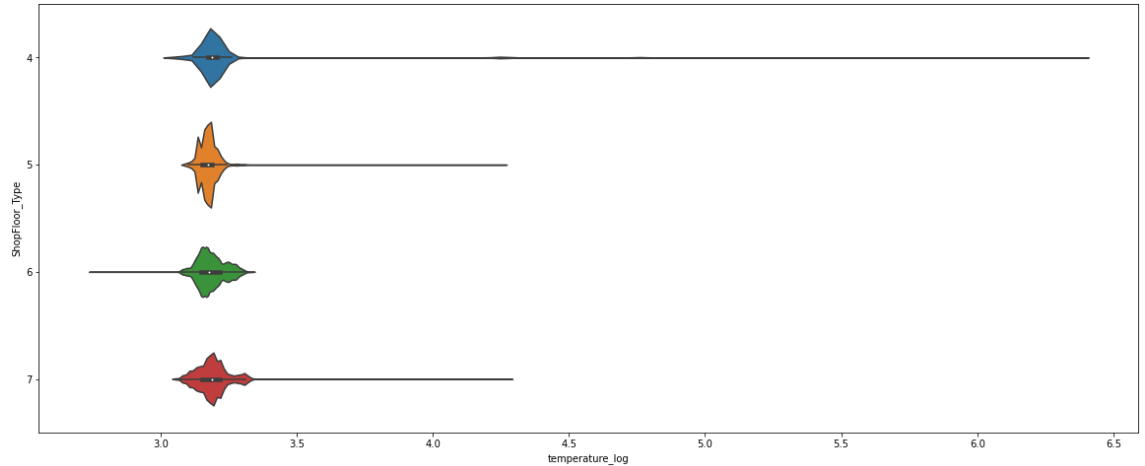
fig**.**set\_size\_inches**(**20**,** 8**)**

#Insight:

# Multi model distributions with large outliers in all except 6XX shopfloors

# Median temperature is pretty much the same in all shop floors

**OUT [14]**



**IN [15]**

#humidity distribution

final\_data**[**'humidity\_log'**]** **=** final\_data**[**'humidity'**].**apply**(lambda** x**:** np**.**log**(**x**+**1**))**

fig **=** plt**.**gcf**()**

sns**.**violinplot**(**x**=** "humidity\_log"**,** y**=**"ShopFloor\_Type"**,** data**=**final\_data**)**

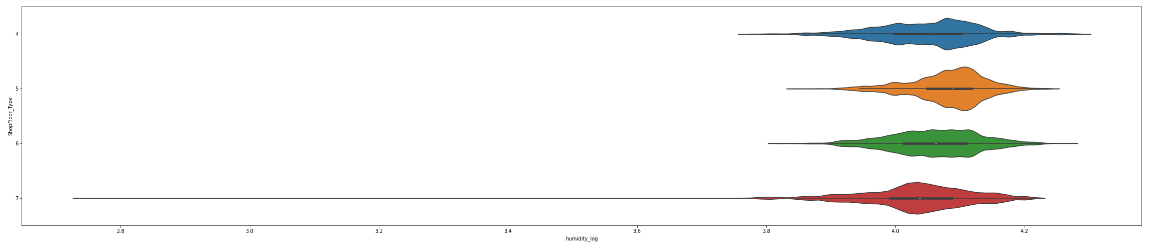
fig**.**set\_size\_inches**(**40**,** 8**)**

#Insight:

# Multi model distributions

#ShopFloor\_Type 7XX, 4XX have similar lower median humidity than the rest

**OUT [15]**



**IN [16]**

# co2 distribution

final\_data**[**'co2\_log'**]** **=** final\_data**[**'co2'**].**apply**(lambda** x**:** np**.**log**(**x**+**1**))**

fig **=** plt**.**gcf**()**

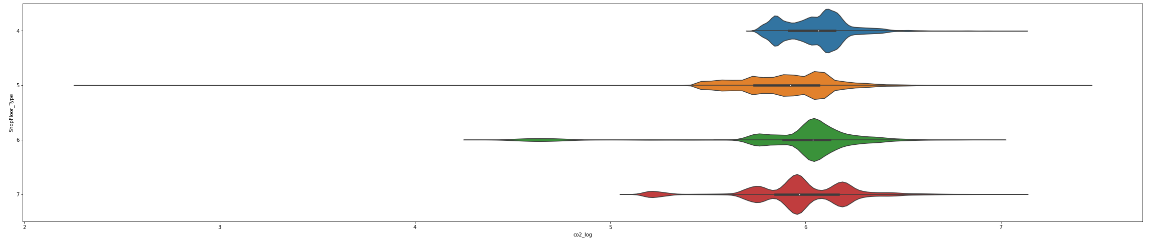
sns**.**violinplot**(**x**=** "co2\_log"**,** y**=**"ShopFloor\_Type"**,** data**=**final\_data**)**

fig**.**set\_size\_inches**(**40**,** 8**)**

# Insight:

# Multi model distributions with large lower level co2 levels in shop floors 5XX

**OUT [16]**



### **Data Exploration challenge 5: Which of the features co2, temperature, humidity and light could potentially help in determining if a shop floor is occupied or not?**

**IN [17]**

# This line creates a binary variable with a value of 1 if the floor is occupied in any capacity & 0 if it is not.

final\_data**[**'Occupied'**]** **=** final\_data**[**'pir'**].**apply**(lambda** x**:** 0 **if** x**==**0 **else** 1**)**

# Insight:

# Median luminosity is higher when shop floors are occupied

# Median temperature is the same with or without occupancy

# Median humidity is the same with or without occupancy

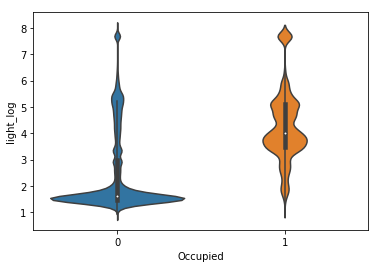
# Median CO2 is higher with occupancy

**IN [18]**

sns**.**violinplot**(**data**=**final\_data**,** y**=**'light\_log'**,** x**=**'Occupied'**)**

**OUT [18]**

<AxesSubplot:xlabel='Occupied', ylabel='light\_log'>

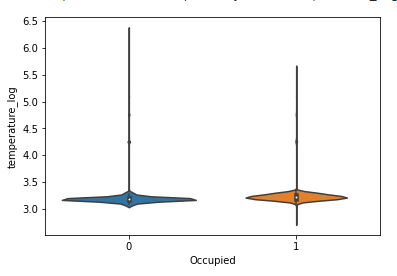


**IN [19]**

sns**.**violinplot**(**data**=**final\_data**,** y**=**'temperature\_log'**,** x**=**'Occupied'**)**

**OUT [19]**

<AxesSubplot:xlabel='Occupied', ylabel='temperature\_log'>

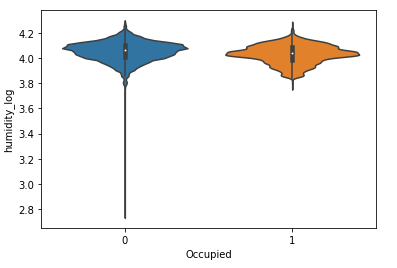


**IN [20]**

sns**.**violinplot**(**data**=**final\_data**,** y**=**'humidity\_log'**,** x**=**'Occupied'**)**

**OUT [20]**

<AxesSubplot:xlabel='Occupied', ylabel='humidity\_log'>

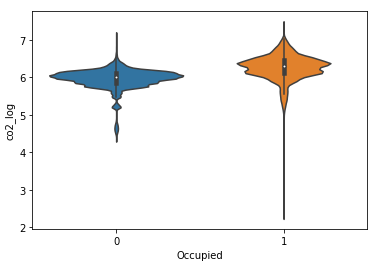


**IN [21]**

sns**.**violinplot**(**data**=**final\_data**,** y**=**'co2\_log'**,** x**=**'Occupied'**)**

**OUT [21]**

<AxesSubplot:xlabel='Occupied', ylabel='co2\_log'>



## Challenge 3: Build a classification model to predict if a shop floor has occupants or not

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data. Once you have trained the model, you can use it to reason over data that it hasn't seen before and make predictions about those data.

### Objectives

* What machine learning models are suitable for this and similar problem statements?
* How to interpret how the machine learning models are tuned to predict future unseen data?

### Tasks

1. What additional features might be interesting for this problem?
2. Choose a model and a metric to build a model to predict occupancy
3. Explain the model and important global features to predict the occupancy

### What additional features might be interesting for this problem?

**IN [22]**

# Based on the data exploration there seems to be two useful features

# 1. Hour - hour of the day

# 2. ShopFloor\_Type - Kind of shop floor

# code to add Hour and ShopFloot\_Type (these have been added during data exploration)

# final\_data['Hour'] = pd.to\_datetime(final\_data['date\_time'],unit='s')

# final\_data['Hour'] = final\_data['Hour'].dt.hour

#

# final\_data['ShopFloor\_Type'] = final\_data['ShopFloor'].str[0]

X **=** final\_data**[[**'co2\_log'**,**'humidity\_log'**,**'temperature\_log'**,**'light\_log'**,**'Hour'**,**'ShopFloor\_Type'**]]** # Adding two additional features Hour and ShopFloor\_Type

**IN [23]**

# We're assuming 0 means no occupants and everything above 0 means occupied

# alternatively, I could leave this target as is, and do regression, however, I don't want # to.

y **=** final\_data**[**'Occupied'**]**

# Convert Hour and Sensor into categories.

# While the features are identified by numbers, the numbers themselves refer to

# categorical (qualitative) data, not numeric (quantitative) data.

X**[**'ShopFloor\_Type'**]** **=** X**[**'ShopFloor\_Type'**].**astype**(**'category'**)**

X**[**'Hour'**]** **=** X**[**'Hour'**].**astype**(**'category'**)**

# Identify numerical and categorical features

categorical\_features **=** **[**'Hour'**,**'ShopFloor\_Type'**]**

numerical\_features **=** **[**'co2\_log'**,**'humidity\_log'**,**'temperature\_log'**,**'light\_log'**]**

**/**anaconda**/**envs**/**azureml\_py36**/**lib**/**python3.6**/**site**-**packages**/**ipykernel\_launcher**.**py**:**6**:** SettingWithCopyWarning**:**

A value **is** trying to be **set** on a copy of a **slice** **from** a DataFrame**.**

Try using **.**loc**[**row\_indexer**,**col\_indexer**]** **=** value instead

See the caveats **in** the documentation**:** http**://**pandas**.**pydata**.**org**/**pandas**-**docs**/**stable**/**user\_guide**/**indexing**.**html#returning-a-view-versus-a-copy

**/**anaconda**/**envs**/**azureml\_py36**/**lib**/**python3.6**/**site**-**packages**/**ipykernel\_launcher**.**py**:**7**:** SettingWithCopyWarning**:**

A value **is** trying to be **set** on a copy of a **slice** **from** a DataFrame**.**

Try using **.**loc**[**row\_indexer**,**col\_indexer**]** **=** value instead

See the caveats **in** the documentation**:** http**://**pandas**.**pydata**.**org**/**pandas**-**docs**/**stable**/**user\_guide**/**indexing**.**html#returning-a-view-versus-a-copy

**import** sys

**Proctor Notes:** Based on the initial data exploration steps and the creativity of the participant, participants may end up with a different set of features than the example above. Participants might choose to characterize time of day based on a typical shop schedule rather than by an hour, include lag variables to convey a relative change in temperature or other sensor measurements. This list is by no means exhaustive.

### **Choose a model and a metric to build a model to predict occupancy**

**IN [24]**

# Data prep for modeling with logistic regression classifier.

**import** random

**from** sklearn**.**model\_selection **import** train\_test\_split

**from** sklearn**.**compose **import** ColumnTransformer

**from** sklearn**.**impute **import** SimpleImputer

**from** sklearn**.**pipeline **import** Pipeline

**from** sklearn**.**preprocessing **import** StandardScaler**,** OneHotEncoder

# Prediction metrics

**from** sklearn**.**metrics **import** accuracy\_score**,** recall\_score**,** precision\_score**,** mean\_absolute\_error

**from** sklearn**.**metrics **import** roc\_curve

**from** sklearn**.**metrics **import** confusion\_matrix

**from** sklearn**.**metrics **import** roc\_auc\_score

# Split data into training set and test set

random**.**seed**(** 101 **)**

X\_train**,** X\_test**,** y\_train**,** y\_test **=** train\_test\_split**(**X**,** y**,** test\_size**=**0.30**,** random\_state**=**0**,** stratify**=**y**)**

# Ensure indices are aligned between X, y after all the slicing and splitting of DataFrames and Series

X\_train **=** X\_train**.**reset\_index**(**drop**=True)**

X\_test **=** X\_test**.**reset\_index**(**drop**=True)**

y\_train **=** y\_train**.**reset\_index**(**drop**=True)**

y\_test **=** y\_test**.**reset\_index**(**drop**=True)**

# Column transformers

numeric\_transformer **=** Pipeline**(**steps**=[**

**(**'imputer'**,** SimpleImputer**(**strategy**=**'most\_frequent'**)),**

**(**'scaler'**,** StandardScaler**())])**

categorical\_transformer **=** Pipeline**(**steps**=[**

**(**'imputer'**,** SimpleImputer**(**strategy**=**'most\_frequent'**)),**

**(**'onehot'**,** OneHotEncoder**(**handle\_unknown**=**'ignore'**))])**

preprocessor **=** ColumnTransformer**(**

transformers**=[**

**(**'num'**,** numeric\_transformer**,** numerical\_features**),**

**(**'cat'**,** categorical\_transformer**,** categorical\_features**)**

**])**

X\_train **=** preprocessor**.**fit\_transform**(**X\_train**)** #use fit\_transform for training data

X\_test **=** preprocessor**.**transform**(**X\_test**)** #use transform for test data

**Proctor Notes:** In this example, for the train-test split, the test set is 30% of the input (the rest is in the train set), the random state of 0 will ensure that the split will be the same each time (ensuring reproducibility), & stratify=y ensures that the distribution of 0 and 1 will be the same in the train & test sets as in the original data. There participants might choose other splits or other methods of model evaluation.

The code above adjusts the values to the normal distribution of each numeric feature by subtracting the mean of each feature and dividing them by the standard deviation. Then the code creates a unique binary column for every categorical feature’s value and sets that column’s value at 1 if the category is equal to that value and 0 if not (e.g., for the Hour feature, there will be 24 binary columns for each hour of the day). This ensures that the hour & floor features are not treated as numeric features by the model. There are likely other valid approaches to data processing and preparing the data for modelling.

**IN [25]**

#Model 1: Logistic Regression

#Logistic Regression Model

**from** sklearn**.**linear\_model **import** LogisticRegression

logit\_model**=** LogisticRegression**(**solver**=**"liblinear"**,** fit\_intercept**=True,** class\_weight**=**'balanced'**)**

logit\_model**.**fit**(**X\_train**,** y\_train**)**

# Get predictions from test data

y\_hat **=** logit\_model**.**predict**(**X\_test**)**

cm **=** confusion\_matrix**(**y\_test**,** y\_hat**)**

**print** **(**'Confusion Matrix:\n'**,**cm**,** '\n'**)**

**print(**'Accuracy:'**,** accuracy\_score**(**y\_test**,** y\_hat**))**

# calculate AUC

y\_scores **=** logit\_model**.**predict\_proba**(**X\_test**)**

auc **=** roc\_auc\_score**(**y\_test**,**y\_scores**[:,**1**])**

**print(**'AUC: ' **+** **str(**auc**))**

**OUT [25]**

Confusion Matrix:

[[123351 28093]

[ 1013 7653]]

Accuracy: 0.818212478920742

AUC: 0.9183563020765286

**Proctor Notes:** Participants may use different metrics to evaluate their models. Specific metrics can be influences by model approach and findings in the data. Some commonly used metrics would be the confusion matrix, F1-Scores, AUC and Accuracy. In the coded example, the Confusion Matrix for the test set also shows that the Recall is 0.8727 and the Precision is 0.2141. The high recall means that the model does well at correctly predicting that the floor is occupied when it is, but out of all times the model predicts that the floor is occupied, it is only occupied in slightly over 21% of those predictions.

This is due to the class imbalance present in the training data with the floor being occupied in only slightly over 5% of all data. This can be rectified by several methods, such as oversampling data from the minority class (Occupancy of 1), adjusting the class weight in the Logistic Regression model, or by adjusting the probability threshold used to decide if a floor is occupied from the default score of 0.5.

The specific values for the metrics will change depending on the specific features created and how the participant has processed their data.

### **Explain which features the indicates are most useful for predicting whether a shop floor is occupied. Explain the relationship between different model features and model predictions.**

**IN [26]**

**def** get\_feature\_names**(**column\_transformer**):**

"""Get feature names from all transformers.

Returns

-------

feature\_names : list of strings

Names of the features produced by transform.

"""

# Remove the internal helper function

#check\_is\_fitted(column\_transformer)

# Turn lookup into function for better handling with pipeline later

**def** get\_names**(**trans**):**

# >> Original get\_feature\_names() method

**if** trans **==** 'drop' **or** **(**

**hasattr(**column**,** '\_\_len\_\_'**)** **and** **not** **len(**column**)):**

**return** **[]**

**if** trans **==** 'passthrough'**:**

**if** **hasattr(**column\_transformer**,** '\_df\_columns'**):**

**if** **((not** **isinstance(**column**,** **slice))**

**and** **all(isinstance(**col**,** **str)** **for** col **in** column**)):**

**return** column

**else:**

**return** column\_transformer**.**\_df\_columns**[**column**]**

**else:**

indices **=** np**.**arange**(**column\_transformer**.**\_n\_features**)**

**return** **[**'x%d' **%** i **for** i **in** indices**[**column**]]**

**if** **not** **hasattr(**trans**,** 'get\_feature\_names'**):**

# >>> Change: Return input column names if no method available

# Turn error into a warning

# For transformers without a get\_features\_names method, use the input

# names to the column transformer

**if** column **is** **None:**

**return** **[]**

**else:**

**return** **[**name **+** "\_\_" **+** f **for** f **in** column**]**

**return** **[**name **+** "\_\_" **+** f **for** f **in** trans**.**get\_feature\_names**()]**

### Start of processing

feature\_names **=** **[]**

# Allow transformers to be pipelines. Pipeline steps are named differently, so preprocessing is needed

**if** **type(**column\_transformer**)** **==** sklearn**.**pipeline**.**Pipeline**:**

l\_transformers **=** **[(**name**,** trans**,** **None,** **None)** **for** step**,** name**,** trans **in** column\_transformer**.**\_iter**()]**

**else:**

# For column transformers, follow the original method

l\_transformers **=** **list(**column\_transformer**.**\_iter**(**fitted**=True))**

**for** name**,** trans**,** column**,** \_ **in** l\_transformers**:**

**if** **type(**trans**)** **==** sklearn**.**pipeline**.**Pipeline**:**

# Recursive call on pipeline

\_names **=** get\_feature\_names**(**trans**)**

# if pipeline has no transformer that returns names

**if** **len(**\_names**)==**0**:**

\_names **=** **[**name **+** "\_\_" **+** f **for** f **in** column**]**

feature\_names**.**extend**(**\_names**)**

**else:**

feature\_names**.**extend**(**get\_names**(**trans**))**

**return** feature\_names

**IN [27]**

**import** sklearn

coefs **=** logit\_model**.**coef\_**[**0**]**

x **=** get\_feature\_names**(**preprocessor**)**

x **=** **[**w**.**replace**(**'x0'**,** 'Hour'**)** **for** w **in** x**]**

x **=** **[**w**.**replace**(**'x1'**,** 'Shopfloor\_Type'**)** **for** w **in** x**]**

x

**OUT [27]**

['num\_\_co2\_log',

'num\_\_humidity\_log',

'num\_\_temperature\_log',

'num\_\_light\_log',

'onehot\_\_Hour\_0',

'onehot\_\_Hour\_1',

'onehot\_\_Hour\_2',

'onehot\_\_Hour\_3',

'onehot\_\_Hour\_4',

'onehot\_\_Hour\_5',

'onehot\_\_Hour\_6',

'onehot\_\_Hour\_7',

'onehot\_\_Hour\_8',

'onehot\_\_Hour\_9',

'onehot\_\_Hour\_10',

'onehot\_\_Hour\_11',

'onehot\_\_Hour\_12',

'onehot\_\_Hour\_13',

'onehot\_\_Hour\_14',

'onehot\_\_Hour\_15',

'onehot\_\_Hour\_16',

'onehot\_\_Hour\_17',

'onehot\_\_Hour\_18',

'onehot\_\_Hour\_19',

'onehot\_\_Hour\_20',

'onehot\_\_Hour\_21',

'onehot\_\_Hour\_22',

'onehot\_\_Hour\_23',

'onehot\_\_Shopfloor\_Type\_4',

'onehot\_\_Shopfloor\_Type\_5',

'onehot\_\_Shopfloor\_Type\_6',

'onehot\_\_Shopfloor\_Type\_7']

**Proctor Notes:** Responsible AI is explainable and here good solutions will focus on the business context and explain the relationship between features and the model. Some features may be more reliable longer term and less sensitive to change (for example, a model too focused on shop hours might be more likely to fail in an off-hours or special case scenario.)

There’s no singular definition of variable importance- sometimes in a regression context a variable could have a small coefficient, but be easy to change vs others with a large coefficient and little variance. The solutions provided by the code example are based on the size of coefficient, however participants could have solutions related to features increasing model performance.

**IN [28]**

# Since feature importance is not built into Logistic regression. We will calculate the Euler number to the power of its coefficient to find the importance.

**import** math

feature\_importance **=** pd**.**DataFrame**(**x**,** columns **=** **[**"feature"**])**

feature\_importance**[**"importance"**]** **=** **pow(**math**.**e**,** coefs**)**

feature\_importance **=** feature\_importance**.**sort\_values**(**by **=** **[**"importance"**],** ascending**=False)**

**from** sklearn**.**linear\_model **import** LogisticRegression

ax **=** feature\_importance**.**plot**.**barh**(**x**=**'feature'**,** y**=**'importance'**)**

plt**.**show**()**

f **=** plt**.**figure**()**

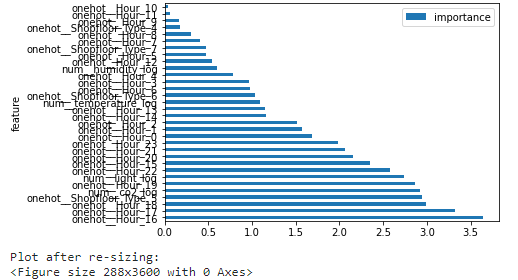
f**.**set\_figwidth**(**4**)**

f**.**set\_figheight**(**50**)**

**print(**"Plot after re-sizing: "**)**

plt**.**show**()**

**OUT [28]**



**IN [29]**

# Insight:

# Tried Logistic Regression Model with AUC of 91%

# Based on global feature importance, the top 3 most important variable in determining the occupancy on a shop floor are Hour 16,17 ,18

## Cleaning up

If you used a compute instance in an Azure Machine Learning workspace to complete the exercise, use these steps to clean up:

* Close all Jupyter notebooks and the Juptyer home page
* In Azure Machine Learning Studio, on the Compute page, select your compute instance and stop it

# Value Proposition

* Develop fluency in sustainability topics especially in carbon and energy space to have meaningful conversations with customers and partners
* Join the community of Microsoft Sustainability Champions and get badged

# Technical Scenarios

* Environmental Monitoring​
* Energy consumption / embodied carbon monitoring and reporting​
* Facilities/equipment operating efficiency optimization​
* Energy source optimization

# Audience

* + - Facilitator/Proctor: ATU, CSU, STU, SMC, OCP, CSA
    - Student: CSA, CSU

# Market Roadmap

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Goals** | **Opportunity Areas** |
| **Carbon /  Energy** |  | * Reduce total & per capita energy consumption * Reduce GHG emissions, Scope 1-3 * Increase contribution of Green Power in total power consumption * Energy & grid management | * Monitor power consumption across operations * Monitor CO2 emissions across operations * Monitor embodied carbon * Monitor & manage carbon in supply chain * Environmental monitoring & reporting * Optimize energy sourcing and supply chain |
| **Ecosystem / Real Estate** |  | * Reduce energy and electricity consumption by facilities and infrastructure * Green Building Ecosystem * Reduce environmental impact on natural ecosystems, including species biodiversity, land / agriculture etc. | * Monitoring building efficiency * Monitor forests, lands, species * Assess environment impact |

Sustainable cloud positions companies to deliver on new commitments: carbon reduction and responsible innovation. Companies have historically driven financial, security, and agility benefits though cloud, but sustainability is becoming an imperative.

|  |  |  |
| --- | --- | --- |
| **44%** | **|** | of CEOs in the United Nations Global Compact – Accenture Strategy CEO study on Sustainability see a net-zero future for their company in the next ten years. |
| **4.7X** | **|** | Between 2013-2019, companies with consistently high environmental, social and governance (ESG) performance enjoyed 4.7X higher operating margins and lower volatility than low ESG performers over the same period. |
| **30-40%** | **|** | Migrations to public cloud result in up to 30-40% total cost of ownership (TCO) savings. |

Drivers like greater workload flexibility, better server utilization rates, and more energy-efficient infrastructure all make public clouds more efficient than enterprise-owned data centers.

# Competitive Landscape

* + - **Salesforce**: <https://www.salesforce.com/products/sustainability-cloud/overview/>
    - **AWS**: <https://sustainability.aboutamazon.com/environment/the-cloud#:~:text=Sustainability%20in%20the%20Cloud%20Amazon%20Web%20Services%20%28AWS%29,100%25%20renewable%20energy%20usage%20for%20our%20global%20infrastructure>.
    - **Google**: <https://sustainability.google/>

# FAQs

1. What are the FAQs participants could have?